IN THE CLAIMS:

1. (Previously Presented) In a signal bearing medium tangibly embodying a program of machine-readable instructions executed by an open source, interoperable IEEE 1394 specification digital device, a method for defining device user interface controls, the method comprising:

from a level two (L2) graphical user interface (GUI), accessing a JAR file; and,

in response to accessing the JAR file, retrieving virtual key information.

2. (Original) The method of claim 1 wherein accessing a JAR file includes accessing a JAR file stored in read only memory (ROM).

3. (Original) The method of claim 2 wherein retrieving virtual key information includes retrieving virtual key information from a JAR file model selected from the group including static classes and data arrays.

4. (Previously Presented) The method of claim 3 wherein retrieving virtual key information in response to accessing the JAR file includes retrieving an application bundled with the virtual key information.

5. (Original) The method of claim 4 in which a first microprocessor machine using a first operating system is included;

the method further comprising:

receiving virtual key information as Java source code;

using a Java compiler, compiling the Java source code into Java virtual machine (JVM) byte codes for the first operating system; and,

using jar tools, archiving the JVM byte codes into a JAR file stored in ROM.

6. (Previously Presented) The method of claim 5 further comprising:

receiving the application as Java source code;

using a Java compiler, compiling the Java source code into Java virtual machine (JVM) byte codes for the first operating system; and,

using jar tools, archiving the JVM byte codes into a JAR file stored in ROM.

7. (Previously Presented) In a signal bearing medium tangibly embodying a program of machine-readable instructions executed by an open source, interoperable IEEE 1394 specification digital device, a method for defining device user interface controls, the method comprising:

from a level two (L2) graphical user interface (GUI) accessing a Java input/output (I/O) ResourceBundle; and,

in response to accessing the ResourceBundle, retrieving virtual key information.

8. (Original) The method of claim 7 wherein accessing the ResourceBundle includes using a ResourceBundle application program interface (API) to specify a property file.

9. (Previously Presented) The method of claim 8 in which a first microprocessor machine using a first operating system is included;

the method further comprising:

maintaining an application in a protocol associated with the first operating system; and,

wherein accessing the ResourceBundle includes using a ResourceBundle API to specify a property file stored in a file system associated with the first microprocessor machine.

10. (Original) The method of claim 9 wherein using a ResourceBundle API to specify a property file stored in the file system includes specifying a property file stored in an input/output (I/O) device selected from the group of storage devices including hard disks and Flash memory.

11. (Original) The method of claim 10 further comprising:

receiving virtual key information as text-based properties attributes in a ResourceBundle property file;

integrating the virtual key information into a table of virtual key characteristics; and, storing the virtual key characteristics table as the ResourceBundle property file.

12. (Previously Presented) In a signal bearing medium tangibly embodying a program of machine-readable instructions executed by an open source, interoperable IEEE 1394 specification digital device, a method for defining device user interface controls, the method comprising:

from a level two (L2) graphical user interface (GUI) calling a Java native interface (JNI);

at the JNI, using Java byte codes to call a storage driver;

from the storage driver, accessing a mapped memory; and,

in response to accessing the mapped memory, retrieving virtual key information.

13. (Original) The method of claim 12 wherein accessing a mapped memory includes accessing a mapped memory stored in an electrically erasable programmable read only memory (EEPROM).

14. (Original) The method of claim 13 wherein retrieving virtual key information includes retrieving virtual key information from mapped memory in a binary format.

15. (Original) The method of claim 14 wherein using Java byte codes to call a storage driver at the JNI includes converting the Java byte code to binary format addresses; and,

wherein accessing a mapped memory from the storage driver includes using the binary format addresses to access ASCII codes stored in the EEPROM.

16. (Original) The method of claim 15 in which a first microprocessor using a first operating system is included;

the method further comprising:

receiving the storage driver as first operating system machine codes; and,

storing the storage driver as machine code.

17. (Original) The method of claim 16 further comprising:

receiving virtual key information as binary format code;

using the storage driver, cross-referencing the virtual key information with EEPROM addresses; and,

storing the virtual key information in the EEPROM as machine code.